Robust Reinforcement Learning with Bayesian Optimisation and Quadrature

Supratik Paul, Konstantinos Chatzilygeroudis, Kamil Ciosek, Jean-Baptiste Mouret, Michael A Osborne, Shimon Whiteson

Journal of Machine Learning Research 21 (2020) 1-31 Submitted 4/18; Published 7/20

Robust Reinforcement Learning with Bayesian Optimisation and Quadrature

Supratik Paul[1] SUPRATIK.PAUL@CS.OX.AC.UK
Konstantinos Chatzilygeroudis[2] KONSTANTINOS.CHATZILYGEROUDIS@INRIA.FR
Kamil Ciosek[1] KAMIL.CIOSEK@CS.OX.AC.UK
Jean-Baptiste Mouret[2] JEAN-BAPTISTE.MOURET@INRIA.FR
Michael A. Osborne[3] MOSB@ROBOTS.OX.AC.UK
Shimon Whiteson[1] SHIMON.WHITESON@CS.OX.AC.UK
[1] *Department of Computer Science, University of Oxford*
Wolfson Building, Parks Road, Oxford OX1 3QD
[2] *Inria; Université de Lorraine; CNRS*
[3] *Department of Engineering Science, University of Oxford*

Editor: Bayesian Optimization Special Issue

Abstract

Bayesian optimisation has been successfully applied to a variety of reinforcement learning problems. However, the traditional approach for learning optimal policies in simulators does not utilise the opportunity to improve learning by adjusting certain environment variables: state features that are unobservable and randomly determined by the environment in a physical setting but are controllable in a simulator. This article considers the problem of finding a robust policy while taking into account the impact of environment variables. We present *alternating optimisation and quadrature* (ALOQ), which uses Bayesian optimisation and Bayesian quadrature to address such settings. We also present *transferable ALOQ* (TALOQ), for settings where simulator inaccuracies lead to difficulty in transferring the learnt policy to the physical system. We show that our algorithms are robust to the presence of significant rare events, which may not be observable under random sampling but play a substantial role in determining the optimal policy. Experimental results across different domains show that our algorithms learn robust policies efficiently.

Keywords: Reinforcement Learning, Bayesian Optimisation, Bayesian Quadrature, Significant rare events, Environment variables

1. Introduction

A key consideration when applying *reinforcement learning* (RL) to a physical setting is the risk and expense of running trials, e.g., while learning the optimal policy for a robot. Another consideration is the robustness of the learned policies. Since it is typically infeasible to test a policy in all contexts, it is difficult to ensure it works as broadly as intended. Fortunately, policies can often be tested in a simulator that exposes key *environment variables*, state features that are unobserved and randomly determined by the environment in a physical setting but are controllable in the simulator. For example, in a hexapod robot, environment variables could describe the state of its legs, which can vary from fully op-

erational to completely broken off, due to damage taken during operation. This article considers how to use environment variables to help learn robust policies.

Although trials in a simulator are cheaper and safer than physical trials, the computational cost of each simulated trial can still be high. The challenge then is to develop algorithms that are sample efficient, i.e., that minimise the number of such trials. In such settings, *Bayesian optimisation* (BO) (Brochu et al., 2010) is a sample-efficient approach that has been successfully applied to RL in multiple domains (Lizotte et al., 2007; Martinez-Cantin et al., 2007, 2009; Cully et al., 2015; Calandra et al., 2015; Pautrat et al., 2018).

A naïve approach would be to randomly sample the environment variable in each trial, so as to estimate expected performance. However, this approach (1) often requires testing each policy in a prohibitively large number of scenarios, and (2) is not robust to *significant rare events* (SREs), i.e., it fails any time there are rare events that substantially affect expected performance. For example, rare localisation errors may mean that a robot is much nearer to an obstacle than expected, increasing the risk of a collision. Since collisions are so catastrophic, avoiding them is key to maximising expected performance, even though the factors contributing to the collision occur only rarely. In such cases, the naïve approach will not see such rare events often enough to learn an appropriate response.

Instead, we present a new approach called *alternating optimisation and quadrature* (ALOQ) (Paul et al., 2018) specifically aimed at learning policies that are robust to these rare events while remaining sample efficient. The main idea is to *actively* select the environment variables (instead of sampling them) in a simulator. We use a *Gaussian process* (GP) (Rasmussen and Williams, 2005) to model returns as a function of *both* the policy and the environment variables and then, at each step, alternately use BO and *Bayesian quadrature* (BQ) (O'Hagan, 1991; Rasmussen and Ghahramani, 2003) to select a policy and environment setting, respectively, to evaluate.

A simulator is only an imperfect representation of reality and policies that are exclusively learnt in simulation can exploit badly modelled aspects of reality and end up performing significantly worse on the physical system. This poses the challenge of bridging the *reality gap* (Jakobi et al., 1995; Jakobi, 1997; Koos et al., 2013), i.e., transferring policies from the simulator to the real system without any significant downgrade in performance. In this article we present an extension to ALOQ, called *transferable ALOQ* (TALOQ), that automatically trades off the cost and accuracy of running a trial on the simulator against running it on the physical system.

We first apply ALOQ to a number of primarily simulated problems, where the reality gap does not pose any significant challenge. Our results demonstrate that ALOQ learns better and faster than multiple baselines. Next, we demonstrate that TALOQ can use an imperfect simulator to learn policies that bridge the reality gap, while requiring significantly fewer trials than if we were to train with ALOQ solely on the robot.[1]

1. This article extends a conference paper (Paul et al., 2018) that introduced ALOQ but required an accurate simulator to ensure transferability to the physical system. This article adds TALOQ, which addresses this issue by actively selecting whether to run a trial on the physical system or the simulator at each iteration. It also adds new experimental results showing that a policy learnt by TALOQ transfers well to the physical system, comparisons of performance of TALOQ to ALOQ and other baselines, and an empirical evaluation of the robustness of TALOQ to the additional hyperparameter it introduces.

2. Related Work

Here we present some of the methods proposed in the literature for optimising problems with environment variables, learning robust policies, and for bridging the reality gap.

2.1. Optimising in the presence of environment variables

Williams et al. (2000) consider a problem setting they call the *design of computer experiments* that is similar to our setting, but does not specifically consider SREs. Their proposed GP-based approach marginalises out the environment variable by alternating between BO and BQ. However, unlike ALOQ, their method is based on the EI acquisition function, which makes it computationally expensive for reasons discussed in Section 5, and is applicable only to discrete environment variables. We include their method as a baseline in our experiments. Our results (in Section 8.2) show that their method is unsuitable for settings with SREs. Further, their method fails even to outperform a baseline that randomly samples the environment variable at each step.

Toscano-Palmerin and Frazier (2018) propose a BO based method for the same setting. They show that their method, Bayesian Quadrature Optimisation, which uses a value of information criterion to select points for evaluation, is more efficient than that of Williams et al. (2000). However, like Williams et al. (2000) they do not explicitly consider SREs, and their method likely to fail without the benefit of the machinery developed in ALOQ (discussed in detail in Section 5) to specifically address the issues raised by SREs.

Krause and Ong (2011) also address optimising performance in the presence of environment variables. However, they address a fundamentally different contextual bandit setting in which the learned policy conditions on the observed environment variable.

2.2. Learning Policies Robust to SREs

Frank et al. (2008) also consider the problems posed by SREs. In particular, they propose an approach based on importance sampling (IS) for efficiently evaluating policies whose expected value may be substantially affected by rare events. While their approach is based on *temporal difference* (TD) methods, we take a BO-based policy search approach. Unlike TD methods, BO is well suited to settings in which sample efficiency is paramount and/or where assumptions (e.g., the Markov property) that underlie TD methods cannot be verified. More importantly, they assume prior knowledge of the SREs, such that they can directly alter the probability of such events during policy evaluation. By contrast, a key strength of ALOQ is that it requires only that a set of environment variables can be controlled in the simulator, without assuming any prior knowledge about whether SREs exist, or about the settings of the environment variables that might trigger them.

Ciosek and Whiteson (2017) also proposed an IS-based algorithm, OFFER, where the setting of the environment variable is gradually changed based on observed trials. Since OFFER, as a policy gradient method, relies on properties of the Markov Decision Process such as the Markov property, it suffers from all the disadvantages mentioned earlier. Also, it can lead to unstable IS estimates if the environment is modified in other ways than by changing the initial state.

Rajeswaran et al. (2017) propose EPOpt for finding robust policies by training on different versions of a simulator. First, multiple instances of the simulator are generated by drawing a random sample of the simulator parameter settings. Trajectories are then sampled from each of these instances and used by a batch policy optimisation algorithm (e.g., TRPO (Schulman et al., 2015)). While ALOQ finds a risk-neutral solution, EPOpt finds a risk-averse one that maximises the *conditional value at risk* (CVaR) by feeding the policy optimisation only the sampled trajectories whose returns are lower than the CVaR. In a risk-neutral setting, EPOpt reduces to the underlying policy optimisation algorithm with trajectories randomly sampled from different instances of the simulator. This approach will not see SREs often enough to learn an appropriate response, as we demonstrate in our experiments.

Pinto et al. (2017) also suggest a method to address the problem of finding robust policies. Their method learns a policy by training in a simulator that is adversarial in nature, i.e., the simulator settings are dynamically chosen to minimise the returns of the policy. This method requires significant prior knowledge to be able to set the simulator settings such that it provides just the right amount of challenge to the policy. Furthermore, it does not consider any settings with SREs.

2.3. Learning policies for physical systems

The methods suggested for learning policies for physical systems address the need for sample efficiency by a variety of means: they can use be model based, use sample efficient optimisation techniques like BO, and incorporate the use of simulators in the learning loop (see Chatzilygeroudis et al. (2019) for a survey). Here we discuss some of these methods.

Levine et al. (2016) and Chebotar et al. (2017) use *guided policy search* (GPS) (Levine and Koltun, 2013) to learn a deep neural net based visuomotor policy for various object manipulation tasks. Yahya et al. (2017) propose a distributed, asynchronous version of GPS to accelerate training and improve generalisation. While GPS has proved to be highly sample efficient, it are unsuitable for our setting. A straightforward application of it ignores the setting of the environment variable, and treats its effect on the transition and rewards as noise. As such the underlying linear-quadratic assumptions of the transition dynamics and rewards are going to be grossly violated as these effects can be very large due to the presence of SREs.

Another approach is to learn a controller in simulation and then fine tune it on the real robot. For example, Lipson and Pollack (2000) learn a design of a robot in simulation, which is then 3D-printed and evolved for a few more steps for fine tuning. Cully et al. (2015) learn a low dimensional behaviour map of diverse controllers in simulation and then perform BO to find the optimal controller on the robot. Inspired by the kernel structure proposed in Poloczek et al. (2017), Marco et al. (2017) develop an entropy search based BO algorithm that together with selecting a policy for evaluation, also selects whether to evaluate the policy on the simulator or the physical system at each iteration.

PILCO (Deisenroth and Rasmussen, 2011) uses data from trials on the physical system to build a model of the environment, and has been shown to achieve remarkable sample efficiency in robot control. Kamthe and Deisenroth (2018) propose a similar approach that maintains data efficiency, while being less computationally intensive and better able to

handle state and control constraints. A similar approach, Black-DROPS (Chatzilygeroudis et al., 2017), is a purely black-box model-based policy search algorithm for robotics and can take advantage of parallel computations. Black-DROPS achieves similar data-efficiency to PILCO. However, like PILCO it scales poorly as the dimensionality of the state/action space increases. A recent extension of Black-DROPS (Chatzilygeroudis and Mouret, 2018) combines nonparametric model learning (using GPs) and model identification in order to scale up to high-dimensional systems.

These methods superficially resemble ALOQ in its use of GPs but the key difference is that they are model based approaches where the GP models the transition dynamics, while ALOQ is model free with the GP modelling the returns as a function of the policy and environment variable. These methods are fundamentally ill suited to our setting. First, they assume that the transition dynamics are Gaussian and can be learned with a few hundred observed transitions, which is often infeasible in more complex environments (i.e., they scale poorly as the dimensionality of the state/action space increases). Second, during training they would be unable to learn a good transition model that conditions on the environment variable since SREs might not be observed often enough under random sampling. Even if they were to learn such a transition model, they would still have to learn a policy that marginalises out the environment variable since it is not observable outside of training. In this case, propagating the uncertainties through the trajectory during training would lead to major violations of the Gaussian assumption when the environment variables can cause SREs.

3. Problem Setting

We assume access to a computationally expensive simulator that takes as input a policy that is parametrised by $\pi \in \Pi$ and environment variable $\theta \in \Theta$ and produces as output the return $f(\pi, \theta) \in \mathbb{R}$, where both Π and Θ belong to some compact sets in $\mathbb{R}^{d_\pi}$ and $\mathbb{R}^{d_\theta}$, respectively. Note that θ refers to the setting of the environment variable, and is independent of the policy whose parameters are given by π. We assume that this simulator is highly accurate and poses little to no reality gap and thus policies learnt on this simulator transfer well to the physical system.

We also assume access to $p(\theta)$, the probability distribution over θ. $p(\theta)$ may be known a priori, or an approximation of it may be available. For example, this could be based on the knowledge of some human expert, or as in the case of a mobile robot it could be the robot's belief about its distance from potential obstacles. While the inferred distribution for $p(\theta)$ based on limited trials may only be a rough approximation of the true distribution, our experiments show that taking account of this approximate distribution can lead to better results than running policy search on some point estimate of θ.

Our objective is to find an optimal policy π^*:

$$\pi^* = \operatorname*{argmax}_\pi \bar{f}(\pi) = \operatorname*{argmax}_\pi \mathbb{E}_\theta[f(\pi, \theta)]. \tag{1}$$

Since evaluating $f(\pi, \theta)$ is expensive, BQ is well suited for computing $\bar{f}(\pi)$, and BO can be a sample efficient framework for finding $\pi^* = \operatorname{argmax}_\pi \bar{f}(\pi)$. We describe BO and BQ in detail in Section 4.

In Section 6, we present a method for a setting in which we relax the assumption that the simulator is accurate. Instead, we assume that the simulator is imperfect but still useful enough that running trials on it is informative about the performance on the physical system. We assume that during training θ can be controlled on the physical system, which is not very restrictive in practice. For example, a broken joint in a robot can be emulated by disabling or limiting the actuator for that joint.

In this case, we define the return $f = f(\pi, \theta, \delta)$, where $\delta \in \{0, 1\}$ is an indicator function that denotes whether the return is from a simulated trial or from the physical system. Our objective is to find the optimal policy π^*:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \bar{f}(\pi) = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_\theta[f(\pi, \theta, 1)], \tag{2}$$

where $\delta = 1$ since we want the policy that maximises returns on the physical system.

4. Background

GPs provide a principled way of quantifying uncertainties associated with modelling unknown functions. A GP is a distribution over functions, and is fully specified by its mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$ (see Rasmussen and Williams (2005) for an in-depth treatment) which encode prior belief about the nature of the function. The prior can be combined with observed values to update the belief about the function in a Bayesian way to generate a posterior distribution.

The prior mean function of the GP is often assumed to be 0 for convenience. A popular choice for the covariance function is the squared exponential which has the form

$$k(\mathbf{x}, \mathbf{x}') = \exp\left\{-\sum_{d=1}^{D}\left(\frac{x_d - x'_d}{l_d}\right)^2\right\}, \tag{3}$$

where D is the dimensionality of x and l_d is the lengthscale associated with the dth dimension. The squared exponential covariance function belongs to the class of stationary functions of the form $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$, which implies that the correlation between the function values of any two points depends only on the distance between them. Informally, a large lengthscale along a particular dimension implies that the function values change relatively slowly along that dimension.

In GP regression, it is assumed that the observed function values $\{f(\mathbf{x}_i)\}_{i=1}^N$ are a sample from a multivariate Gaussian distribution. The prediction for a new point $\mathbf{x}^*$ is connected with the observations through the mean and covariance functions. By conditioning on the observed data, this can be computed analytically as a Gaussian $\mathcal{N}\Big(\mathbb{E}\big(f(\mathbf{x}^*)\big), \mathrm{Cov}\big(f(\mathbf{x}^*)\big)\Big)$:

$$\mathbb{E}\big(f(\mathbf{x}^*)\big) = k(\mathbf{x}^*, \mathbf{X})(\mathbf{K} + \sigma^2_{\text{noise}}\mathbf{I})^{-1} f(\mathbf{X}) \tag{4a}$$

$$\mathrm{Cov}\big(f(\mathbf{x}^*)\big) = k(\mathbf{x}^*, \mathbf{x}^*) - k(\mathbf{x}^*, \mathbf{X})(\mathbf{K} + \sigma^2_{\text{noise}}\mathbf{I})^{-1} k(\mathbf{X}, \mathbf{x}^*), \tag{4b}$$

where $\mathbf{X}$ denotes the vector of observed inputs, $f(\mathbf{X})$ the vector of corresponding function values, and $\mathbf{K}$ is the matrix with entries $k(\mathbf{x}_i, \mathbf{x}_j), i, j = 1, 2, ..., N$.

4.1. Bayesian Optimisation

Bayesian optimisation is a framework for optimising black-box gradient-free functions that are expensive to evaluate. It works by iteratively building a surrogate model of the function based on observed data, and using this surrogate model to actively select the next point to query. The property of generating estimates of the uncertainty associated with any prediction makes GPs particularly suited as the surrogate model. Once the surrogate GP model has been built, BO uses an *acquisition function* to guide the search and balance exploitation (searching the space expected to have the optimum) and exploration (searching the space which has not been explored well) to choose the next point to be queried.

Concretely, at iteration n, BO fits a GP to the observed data $\mathcal{D}_{1:n-1} = \{(\mathbf{x}_i, f(\mathbf{x}_i))\}_{i=1}^{n-1}$. The next point for evaluation is then set as $\mathbf{x}_n = \operatorname{argmax}_{\mathbf{x}} \alpha(\mathbf{x})$, where $\alpha(\mathbf{x})$ is the acquisition function. Two commonly used acquisition functions are *expected improvement* (EI) (Močkus, 1975; Jones et al., 1998) and *upper confidence bound* (UCB) (Cox and John, 1992, 1997). Defining $\mathbf{x}^+$ as the current optimal evaluation, i.e., $\mathbf{x}^+ = \operatorname{argmax}_{\mathbf{x}_i} f(\mathbf{x}_i)$. EI seeks to maximise the expected improvement over the current optimum,

$$\alpha_{EI}(\mathbf{x}) = \mathbb{E}[I(\mathbf{x})], \text{ where } I(\mathbf{x}) = \max\{0, f(\mathbf{x}) - f(\mathbf{x}^+)\}. \tag{5}$$

By contrast, UCB does not depend on $\mathbf{x}^+$ but directly incorporates the uncertainty in the prediction by defining an upper bound:

$$\alpha_{\text{UCB}}(\mathbf{x}) = \mu(\mathbf{x}) + \kappa\sigma(\mathbf{x}), \tag{6}$$

where κ controls the exploration-exploitation tradeoff.

4.2. Bayesian Quadrature

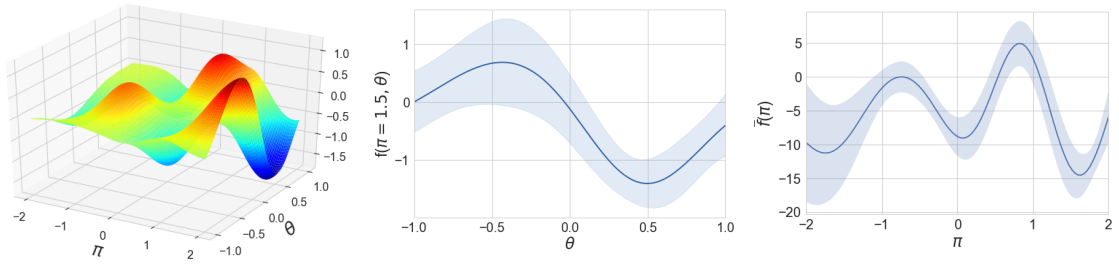
Bayesian quadrature (O'Hagan, 1991; Rasmussen and Ghahramani, 2003) is a sample-efficient technique for computing integrals of the form $\bar{f} = \int f(\mathbf{x})p(\mathbf{x})d\mathbf{x}$, where $p(\mathbf{x})$ is a probability distribution. Using GP regression to compute the prediction for any $f(\mathbf{x})$ given some observed data, $\bar{f}$ is a Gaussian whose mean and variance can be computed analytically for particular choices of the covariance function and $p(\mathbf{x})$ (Briol et al., 2015). If no analytical solution exists, we can approximate the mean and variance via Monte Carlo quadrature by sampling the predictions of various $f(\mathbf{x})$.

Given some observed data $\mathcal{D}$, we can also devise acquisition functions for BQ to actively select the next point $\mathbf{x}^*$ for evaluation. A natural objective here is to select $\mathbf{x}$ that minimises the posterior variance of $\bar{f}$ after observing $f(\mathbf{x})$ (Osborne et al., 2012),

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} \mathbb{V}(\bar{f}|\mathcal{D}, \mathbf{x}). \tag{7}$$

Since $\mathbb{V}(\bar{f}|\mathcal{D}, \mathbf{x})$ is a linear combination of the predictive variances $\mathbb{V}(f(\mathbf{x})|\mathcal{D}, \mathbf{x})$ (4b), which do not depend on $f(\mathbf{x})$, $\mathbb{V}(\bar{f}|\mathcal{D}, \mathbf{x})$ does not depend on $f(\mathbf{x})$ either and is thus computationally feasible to evaluate. *Uncertainty sampling* (Settles, 2010) is an alternative acquisition function that chooses the $\mathbf{x}^*$ with the maximum posterior variance:

$$\mathbf{x}^* = \operatorname*{argmax}_{\mathbf{x}} \mathbb{V}(f(\mathbf{x})|\mathcal{D}). \tag{8}$$



(a) (b) (c)

Figure 1: ALOQ models the return f as a function of (π, θ); (a) the predicted mean based on some observed data; (b) the predicted return of $\pi = 1.5$ for different θ, together with the uncertainty associated with them; (c) given $p(\theta)$, ALOQ marginalises out θ and computes $\bar{f}(\pi)$ and its associated uncertainty, which is used to actively select π.

Although simple and computationally cheap, it is not the same as reducing uncertainty about $\bar{f}$ since evaluating the point with the highest prediction uncertainty does not necessarily lead to the maximum reduction in the uncertainty of the estimate of the integral.

Monte Carlo (MC) quadrature simply samples $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N)$ from $p(\mathbf{x})$ and estimates the integral as $\bar{f} \approx \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}_i)$. This typically requires a large N and so is less sample efficient than BQ: it should only be used if f is cheap to evaluate. The many merits of BQ over MC, both philosophically and practically, are discussed by O'Hagan (1987) and Hennig et al. (2015). For ALOQ and TALOQ, we use an active Bayesian quadrature scheme (i.e., selecting points according to an acquisition function), inspired by the empirical improvements offered by those of Osborne et al. (2012) and Gunter et al. (2014).

5. ALOQ

The key idea behind ALOQ is to model the return as a GP with inputs (π, θ), and use BO to optimise the policy while using BQ to marginalise out the effect of the environment variable on returns. This is illustrated in Figure 1.

Concretely, at each iteration l, ALOQ fits a GP to data set $\mathcal{D}_{1:l-1} = \{((\pi_i, \theta_i), f(\pi_i, \theta_i))\}_{i=1}^{l-1}$. Note that both π and θ are inputs to the GP. Next, it uses a modified UCB acquisition function:

$$\alpha_{\text{ALOQ}}(\pi) = \mu(\bar{f}(\pi) \mid \mathcal{D}_{1:l-1}) + \kappa\sigma(\bar{f}(\pi) \mid \mathcal{D}_{1:l-1}), \tag{9}$$

to set $\pi_l = \text{argmax}_\pi \, \alpha_{\text{ALOQ}}(\pi)$. For discrete θ with support $\{\theta_1, \theta_2, \ldots, \theta_{N_\theta}\}$, the estimate of the mean μ and variance σ^2 for $\bar{f}(\pi) \mid \mathcal{D}_{1:l-1}$ is:

$$\mu = \frac{1}{N_\theta} \sum_{i=1}^{N_\theta} \mathbb{E}[f(\pi, \theta_i) | \mathcal{D}_{1:l-1}] \tag{10a}$$

$$\sigma^2 = \frac{1}{N_\theta^2} \sum_{i=1}^{N_\theta} \sum_{j=1}^{N_\theta} \text{Cov}[f(\pi, \theta_i) | \mathcal{D}_{1:l-1}, f(\pi, \theta_j) | \mathcal{D}_{1:l-1}], \tag{10b}$$

where $f(\pi, \theta)$ is the prediction from the GP with mean and covariance computed using (4). For continuous θ, we apply Monte Carlo quadrature. Although this requires sampling a large number of θ and evaluating the corresponding $f(\pi, \theta) \mid \mathcal{D}_{1:l-1}$, it is feasible since we evaluate $f(\pi, \theta) \mid \mathcal{D}_{1:l-1}$, not from the expensive simulator, but from the computationally cheaper GP. This can be viewed as performing policy evaluation in our approach, i.e., estimating the expected return of a given π after marginalising out the effect of θ.

Although it is possible to define an EI-based acquisition function: $\alpha = \mathbb{E}_{\bar{f}(\pi)|\mathcal{D}_{1:l-1}}[I(\pi)]$, where $I(\pi) = \max\{0, \bar{f}(\pi) - \bar{f}(\pi^+)\}$, as an alternative to the UCB based acquisition function, it is prohibitively expensive to compute in practice. The stochastic $\bar{f}(\pi^+) \mid \mathcal{D}_{1:l-1}$ renders this analytically intractable. Approximating it using Monte Carlo sampling would require performing predictions on $(l-1) \times N_\theta$ points, i.e., all the $(l-1)$ observed π's paired with all the N_θ possible settings of the environment variable, which is infeasible even for moderate $(l-1)$ as the computational complexity of GP predictions scales quadratically with the number of predictions.

Once π_l is chosen, ALOQ uses a BQ acquisition function to select θ_l. Since the presence of SREs leads to high variance in the returns associated with any given policy, it is critical to minimise the uncertainty associated with our estimate of the expected return. Hence, ALOQ selects $\theta_l \mid \pi_l$ by minimising the posterior variance of $\bar{f}(\pi_l)$:

$$\theta_l | \pi_l = \operatorname*{argmin}_{\theta} \mathbb{V}(\bar{f}(\pi_l) | \mathcal{D}_{1:l-1}, \pi_l, \theta). \tag{11}$$

As an alternative, we also tried uncertainty sampling in our experiments. Unsurprisingly, it performed worse since it is not as good at reducing the uncertainty associated with the expected return of a policy, as discussed in Section 4.2.

After (π_l, θ_l) is selected, ALOQ evaluates it on the simulator and updates the GP with the new datapoint $((\pi_l, \theta_l), f(\pi_l, \theta_l))$. Our estimate of π^* is thus:

$$\hat{\pi}^* = \operatorname*{argmax}_{\pi} \mathbb{E}[\bar{f}(\pi) | \mathcal{D}_{1:l}]. \tag{12}$$

5.1. Beta Warping

Although the approach described so far actively selects π and θ through BO and BQ, it is unlikely to perform well in practice. A key observation is that the presence of SREs, which we seek to address with ALOQ, implies that the scale of f varies considerably, e.g., returns in case of collision versus no collision. This nonstationarity cannot be modelled with our stationary kernel. Therefore, we must transform the inputs to ensure stationarity of f. In particular, we employ *beta warping*, i.e., we transform the inputs using beta CDFs with parameters (α, β) (Snoek et al., 2014). The CDF of the beta distribution on the support $0 < x < 1$ is given by:

$$\text{BetaCDF}(x, \alpha, \beta) = \int_0^x \frac{u^{\alpha-1}(1-u)^{\beta-1}}{B(\alpha, \beta)} \mathrm{d}u, \tag{13}$$

where $B(\alpha, \beta)$ is the beta function. The beta CDF is particularly suitable for our purpose as it can model a variety of warpings based on the settings of only two parameters (α, β). ALOQ transforms each dimension of π and θ independently and treats the corresponding (α, β) as hyperparameters. In the rest of this article, we assume that we are working with the transformed inputs.

Algorithm 1 ALOQ

input A simulator that outputs $f = f(\pi, \theta)$, initial data set $\mathcal{D}_{1:l}$, the maximum number of trials L, and a GP prior.

1: **for** $n = l+1, l+3, ..., L-1$ **do**
2: Update the beta warping parameters and transform the inputs.
3: Update the GP to condition on the (transformed) data set $\mathcal{D}_{1:n-1}$
4: Use (10) to estimate $p(\bar{f}|\mathcal{D}_{1:n-1})$
5: Use the BO acquisition function (9) to select $\pi_n = \text{argmax}_\pi \, \alpha_{\text{ALOQ}}(\pi)$
6: Use the BQ acquisition function (11) to select $\theta_n|\pi_n$
7: Perform a simulator call with (π_n, θ_n) to obtain $f(\pi_n, \theta_n)$ and update $\mathcal{D}_{1:n-1}$ to $\mathcal{D}_{1:n}$
8: Find $\hat{\pi}^* = \text{argmax}_{\pi_i} \, \bar{f}(\pi_i)|\mathcal{D}_{1:n}$ and $\theta^*|\hat{\pi}^*$ using the BQ acquisition function (11).
9: Perform a simulator call with $(\hat{\pi}^*, \theta^*)$ to obtain $f(\hat{\pi}^*, \theta^*)$ and update $\mathcal{D}_{1:n}$ to $\mathcal{D}_{1:n+1}$
10: **end for**

output $\pi^* = \text{argmax}_{\pi_i} \, \bar{f}(\pi_i) \mid \mathcal{D}_{1:L} \quad i = 1, 2, ..., L$

5.2. Intensification

While the resulting algorithm should be able to cope with SREs, the $\hat{\pi}^*$ that it returns at each iteration may still be poor, since our BQ evaluation of $\bar{f}(\pi)$ leads to a noisy approximation of the true expected return. This is particularly problematic in high dimensional settings. To address this, *intensification* (Bartz-Beielstein et al., 2005; Hutter et al., 2009), i.e., re-evaluation of selected policies in the simulator, is essential. Therefore, ALOQ performs two simulator calls at each timestep. In the first evaluation, (π_l, θ_l) is selected via the BO/BQ scheme described above. In the second stage, $(\hat{\pi}^*, \theta^*)$ is evaluated, where $\hat{\pi}^* \in \pi_{1:l}$ is selected using (12) and $\theta^*|\hat{\pi}^*$ using the BQ acquisition function (11).

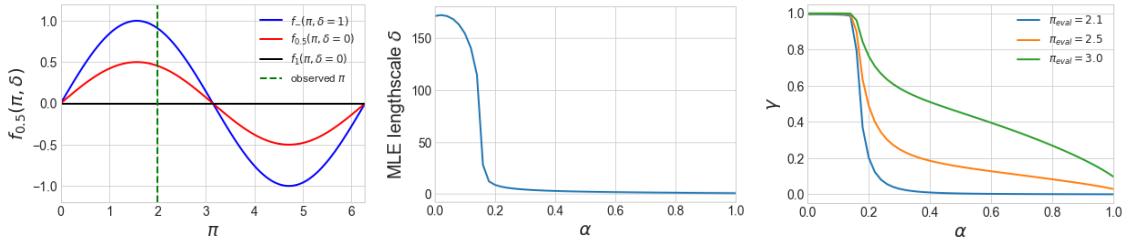
6. TALOQ

ALOQ assumes access to a reliable simulator. In this section, we consider the setting where this assumption does not hold and instead there exists a reality gap between the simulator and the physical system. In this case, at each iteration we face the additional choice to perform the evaluation on the simulator or on the physical system, which is even more expensive. We present a variant of ALOQ called *transferable ALOQ* (TALOQ) that addresses this setting.

6.1. BO with Reality Gap

Before presenting the full TALOQ method, we first formalise the problem setting and describe a simplified BO approach for coping with the reality gap but without considering environment variables. We extend this method to TALOQ in Section 6.2.

Let $\delta \in \{0, 1\}$ denote the simulator or physical system respectively. Assume that at each iteration we can choose to query the simulator or the physical system to obtain $f(\pi, \delta)$ for an input π. Our objective is to find $\pi^* = \text{argmax}_\pi \, f(\pi, \delta = 1)$. In this setting we can model f as a GP with inputs (π, δ). Observing the return of any π in the simulator gives us some information about the corresponding return on the physical system. Since the lengthscale



(a) (b) (c)

Figure 2: An illustrative example: (a) true and simulated function values for different α; (b) the MLE of the lengthscale for δ wrt α; (c) γ for different settings of α for different π_{eval}.

for any dimension encodes the similarity of the function values along that dimension, we can use the lengthscale for δ to quantify how well the simulator reflects reality. If the reality gap is small the lengthscale should be long, and vice versa.

We develop a two-stage algorithm: at iteration l, first select π_l for evaluation, and then $\delta_l|\pi_l$. π_l can be selected by maximising the UCB acquisition function:

$$\alpha(\pi) = \mu(f(\pi, \delta = 1)) + \kappa\sigma(f(\pi, \delta = 1)), \tag{14}$$

where $\delta = 1$ since our objective is to maximise the performance on the physical system.

We define the relative reduction in uncertainty by evaluating π_l in simulation vs. on the physical system:

$$\gamma = \frac{\mathbb{V}[f(\pi_l, 1)|\mathcal{D}_{1:l-1}] - \mathbb{V}[f(\pi_l, 1)|\mathcal{D}_{1:l-1}, \pi_l, \delta_l = 0]}{\mathbb{V}[f(\pi_l, 1)|\mathcal{D}_{1:l-1}] - \mathbb{V}[f(\pi_l, 1)|\mathcal{D}_{1:l-1}, \pi_l, \delta_l = 1]}, \tag{15}$$

and set $\delta_l = 0$ if $\gamma > k$, and $\delta_l = 1$ otherwise. Here $k \in [0, 1]$ is a hyperparameter that should be set based on the relative cost of running physical trials against simulated trials; a large k encourages more physical evaluations, and vice versa. This formulation of γ as a ratio of the relative reduction in variance ensures that it is less problem dependent than a formulation with absolute reduction.

We illustrate this with an example in Figure 2. Let $f_\alpha(\pi, \delta) = (1 - \alpha\mathbb{1}_{\delta=0}) \sin \pi$. Here $\alpha \in [0, 1]$ is a proxy for the reality gap: $\alpha = 1$ corresponds to an uninformative simulator, while $\alpha = 0$ to a perfect simulator with no reality gap. Note that $f_\alpha(\pi, 1)$ is independent of α and hence labelled $f_-(\pi, 1)$ in Figure 2a. Let $f_\alpha(2, 0)$ and $f_\alpha(2, 1)$ be observed. For each setting of α, we can fit a GP to these two observed points and check how the learnt lengthscale varies with α. This is shown in Figure 2b. Note how larger reality gap corresponds to shorter lengthscale. Finally, lets assume that our acquisition function (15) suggests π_{eval} for evaluation. Figure 2c shows how γ varies with α (and hence the lengthscale for δ) for three such π_{eval}. This shows how the relative reduction in uncertainty drops as the reality gap increases, but this drop is much sharper for π_{eval} close to already observed points. Thus our approach has a tendency to make use of the simulator more for unexplored regions, while using reality to fine tune in well explored regions.

Algorithm 2 TALOQ

input A simulator and physical system that outputs $f = f(\pi, \theta)$, initial data set $\mathcal{D}_{1:l}$, the maximum number of trials L, and a GP prior.

for $n = l+1, l+3, ..., L-1$ **do**
  Update the beta warping parameters and transform the inputs.
  Update the GP to condition on the (transformed) data set $\mathcal{D}_{1:n-1}$
  Use (10) to estimate $p(\bar{f}|\mathcal{D}_{1:n-1})$
  Use the BO acquisition function (9) to select $\pi_n = \text{argmax}_\pi \, \alpha_{\text{ALOQ}}(\pi)$
  Compute γ as per (16) and set $\delta_n = 1$ if $\gamma > k$, and $\delta_n = 0$ otherwise.
  Use the BQ acquisition function (17) to select $\theta_n|\pi_n, \delta_n$
  Perform an evaluation with $(\pi_n, \theta_n, \delta_n)$ to obtain $f(\pi_n, \theta_n, \delta_n)$ and update $\mathcal{D}_{1:n-1}$ to $\mathcal{D}_{1:n}$
  Find $\hat{\pi}^* = \text{argmax}_{\pi_i} \, \bar{f}(\pi_i)|\mathcal{D}_{1:n}$
  Compute γ as per (16) and set $\delta^* = 1$ if $\gamma > k$, and $\delta^* = 0$ otherwise.
  Use the BQ acquisition function (17) to select $\theta^*|\hat{\pi}^*, \delta^*$
  Perform a second evaluation with $(\hat{\pi}^*, \theta^*, \delta^*)$ to obtain $f(\hat{\pi}^*, \theta^*, \delta^*)$ and update $\mathcal{D}_{1:n}$ to $\mathcal{D}_{1:n+1}$
end for

output $\pi^* = \text{argmax}_{\pi_i} \, \bar{f}(\pi_i) \mid \mathcal{D}_{1:L} \quad i = 1, 2, ..., L$

6.2. Complete TALOQ Method

In our setting with environment variables, our objective is to find $\pi^* = \text{argmax}_\pi \, \bar{f}(\pi, 1)$ where $\bar{f}(\pi, \delta = 1) = \mathbb{E}_\theta[f(\pi, \theta, \delta = 1)]$, and we can only evaluate $f(\pi, \theta, \delta)$. To address this, TALOQ combines the above selection method for $\delta|\pi$ with ALOQ.

First, we model the return as a GP with three inputs (π, θ, δ). Then, at iteration l, π_l is selected with the acquisition function presented in (9) with the additional condition that $\delta = 1$. Next, we modify the relative reduction in uncertainty for our acquisition function for $\delta_l|\pi_l$ to take into account that θ is yet to be selected:

$$\gamma = \frac{\mathbb{V}[\bar{f}(\pi_l, 1)|\mathcal{D}_{1:l-1}] - \text{argmin}_\theta \, \mathbb{V}[\bar{f}(\pi_l, 1)|\mathcal{D}_{1:l-1}, \pi_l, \theta, \delta_l = 0]}{\mathbb{V}[\bar{f}(\pi_l, 1)|\mathcal{D}_{1:l-1}] - \text{argmin}_\theta \, \mathbb{V}[\bar{f}(\pi_l, 1)|\mathcal{D}_{1:l-1}, \pi_l, \theta, \delta_l = 1]}, \tag{16}$$

and select δ_l based on the hyperparameter k as earlier. Finally, we select $\theta_l|(\pi_l, \delta_l)$ using the BQ acquisition function given in (11) with the slight modification that the conditioning set now includes δ_l:

$$\theta_l|\pi_l, \delta_l = \underset{\theta}{\text{argmin}} \, \mathbb{V}[\bar{f}(\pi_l)|\mathcal{D}_{1:l-1}, \pi_l, \theta, \delta_l]. \tag{17}$$

Marco et al. (2017) propose a related acquisition function for δ for the simplified BO setting described above. However, because it is entropy based, its application to TALOQ is not straightforward, as it would require first integrating over θ. By contrast, our selection criteria for δ is easy to implement and follows the same variance reduction principle as the BQ aspect of selecting θ.

7. Properties of ALOQ and TALOQ

Existing convergence guarantees for BO using α_{UCB} (Srinivas et al., 2010) hold only for cases with fixed hyperparameters. Similarly, existing convergence guarantees for BQ (Kanagawa et al., 2016; Briol et al., 2015) apply only to methods that do not actively select points. Since our methods rely on beta warping, active selection of quadrature points (whether for simulated or real evaluation in TALOQ), and the hyperparameters of the GP are updated after each iteration, these convergence guarantees for BO and BQ cannot be applied to ALOQ and TALOQ. However, we expect such active selection to only improve the rate of convergence of our algorithms over passive versions, and our empirical results in Section 8.2 show that in practice our methods efficiently optimise policies in the presence of SREs across a variety of tasks.

The computational complexity of ALOQ and TALOQ is dominated by an $\mathcal{O}(l^3)$ matrix inversion, where l is the sample size of the data set $\mathcal{D}$. This cubic scaling is common to all BO methods involving GPs. The BQ integral estimation in each iteration requires only GP predictions, which are $\mathcal{O}(l^2)$.

8. Experimental Results

In this section, we empirically compare ALOQ and TALOQ to a number of baselines. Details about the experimental setup, including the environments, choice of kernels, and treatment of hyperparameters amongst others are presented in the Supplementary Materials.

8.1. Baseline Methods

A naïve approach is to directly apply a policy search algorithm (for example, BO, REINFORCE (Williams, 1992), or Trust Region Policy Optimisation (TRPO) (Schulman et al., 2015)) while treating the variability in returns caused by the environment variable as noise.

In particular, we can apply BO directly on $\bar{f}(\pi) = \mathbb{E}_\theta[f(\pi, \theta)]$ and attempt to estimate π^*. Formally, this approach models $\bar{f}$ as a GP with a zero mean function and a suitable covariance function $k(\pi, \pi')$. Since f is expensive to evaluate, at the lth BO iteration only one $f(\pi_l, \theta_l)$ with $\theta_l \sim p(\theta)$ is evaluated in the simulator and $\bar{f}(\pi_l)$ is approximated by this one sample Monte Carlo estimate. This approach will almost surely fail since the estimates of $\bar{f}(\pi)$ are extremely noisy, especially in the presence of SREs.

For policy gradient methods like REINFORCE and TRPO, this translates to sampling a batch of trajectories from the environment at each iteration, and then using the observed returns to approximate gradient ascent on the policy. In general these methods suffer from high variance in the gradient estimates, leading to slow learning (Glynn, 1990; Peters and Schaal, 2006). In our setting with SREs, this problem is compounded by the variance due to θ, which is not explicitly considered while approximating the gradient. Furthermore, if SREs are not observed due to the random sampling from $p(\theta)$, these methods can converge to a highly suboptimal policy. While these methods are not particularly sample efficient and thus not suitable for our setting, we include them in our experiments for completeness.

We also compare ALOQ to several other baselines. The complete list of baselines is 1) the *naïve* method described above with BO, REINFORCE, and TRPO as the underlying policy search algorithms; 2) the method of Williams et al. (2000), which we refer to as

WSN. To show the importance of each component of ALOQ, we also perform experiments with ablated versions, namely: 1) *random quadrature ALOQ* (RQ-ALOQ), in which θ is sampled randomly from $p(\theta)$ instead of being chosen actively; 2) *unwarped ALOQ*, which does not perform beta warping of the inputs; and 3) *one-step ALOQ*, which does not use intensification.

8.2. ALOQ Experiments

To evaluate ALOQ, we applied it to 1) artificial test functions, including those used by Williams et al. (2000), who consider a setting very similar to ours, but without any potential SREs, 2) a simulated robot arm control task, including a variation where $p(\theta)$ is not known a priori but must be inferred from data, and 3) a hexapod locomotion task. All plotted results are the median of 20 independent runs.

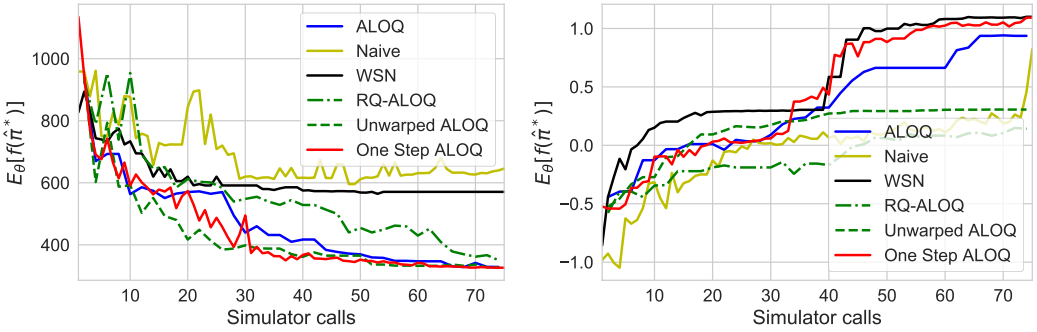
8.2.1. Artificial Test Functions

We begin with modified versions of the *Branin* and *Hartmann 6* test functions used by Williams et al. (2000). The modified Branin test function is a four-dimensional problem, with two dimensions treated as discrete environment variables with a total of 12 support points, while the modified Hartmann 6 test function is six-dimensional with two dimensions treated as environment variables with a total of 49 support points. See Williams et al. (2000) for the mathematical formulation of these functions.

The performance of the algorithms on the two functions is presented in Figure 3. In the Branin function, ALOQ, RQ-ALOQ, unwarped ALOQ, and one-step ALOQ all substantially outperform WSN. WSN performs better in the Hartmann 6 function as it does not get stuck in a local maximum. However, it still cannot outperform one-step ALOQ. Note that ALOQ slightly underperforms one-step ALOQ. This is not surprising: since the problem does not have SREs, the intensification procedure used by ALOQ does not yield any significant benefit.

Figure 4 plots in log scale the total runtime of each algorithm. WSN takes significantly longer than ALOQ or the other baselines, and shows a clear increasing trend. The slow runtime of WSN is as expected for the reasons mentioned in Section 2. However, its failure to outperform RQ-ALOQ is surprising as these are the test problems Williams et al. use in their own evaluation. However, they never compared WSN to these (or any other) baselines. Consequently, they never validated the benefit of modelling θ explicitly, much less selecting it actively. In retrospect, these results make sense because the function is not characterised by significant rare events and there is no other a priori reason to predict that simpler methods will fail.

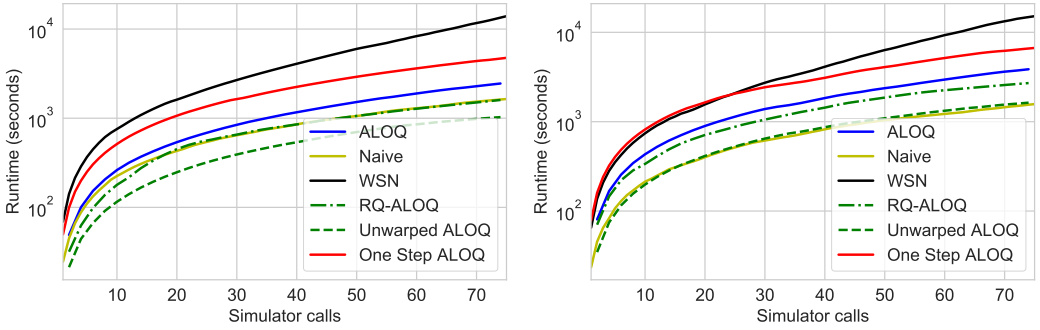
These results underscore the fact that a meaningful evaluation must include a problem with SREs, as such problems do demand more robust methods. To create such an evaluation, we formulated two test functions, F-SRE1 and F-SRE2, that are characterised by significant



(a) Branin (min) - expected value of $\hat{\pi}^*$ (lower is better)

(b) Hartmann 6 (max) - expected value of $\hat{\pi}^*$ (higher is better)

Figure 3: Comparison of performance of all methods on the modified Branin and Hartmann 6 test functions used by Williams et al..



(a) Branin (min)

(b) Hartmann 6 (max)

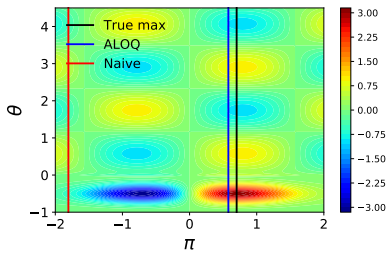
Figure 4: Comparison of runtime of all methods on the modified Branin and Hartmann 6 test function used by Williams et al..

rare events. For $\pi \in [-2, 2]$, F-SRE1 is defined as:

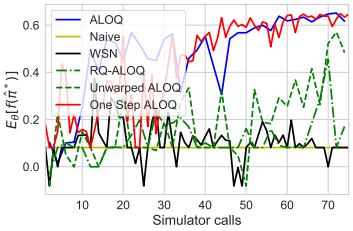
$$\begin{aligned} f_{F-SRE1}(\pi, \theta) =& 75\pi \exp(-\pi^2 - (4\theta + 2)^2) \\ &+ \sin(2\pi)\sin(2.7\theta), \\ \text{with } \; p(\theta = \theta_j) =& \begin{cases} 0.47\% & \text{for } \theta_j = -1.00, -0.95, ..., 0.00 \\ 1.0\% & \text{for } \theta_j = 0.05, 0.10, ..., 4.50. \end{cases} \end{aligned} \tag{18}$$

For $\pi \in [-2, 2]$, F-SRE2 is defined as:

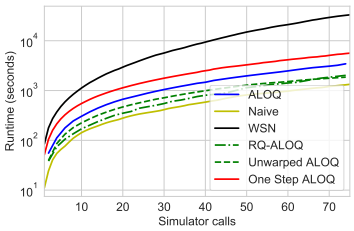
$$\begin{aligned} f_{F-SRE2}(\pi, \theta) =& \sin^2 \pi + 2\cos\theta \\ &+ 200\cos(2\pi)(0.2 - \min(0.2, |\theta|)), \\ \text{with } \; p(\theta = \theta_j) =& \begin{cases} 1.2\% & \text{for } \theta_j = -1.00, -0.98..., -0.22 \\ 0.2\% & \text{for } \theta_j = -0.20, -0.18, ..., 0.20 \\ 1.2\% & \text{for } \theta_j = 0.22, 0.24..., 1.00. \end{cases} \end{aligned} \tag{19}$$



(a) F-SRE1

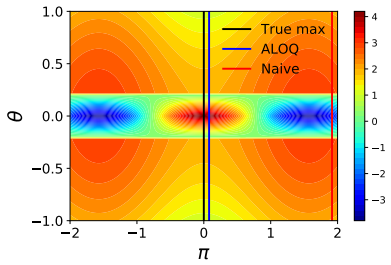


(b) F-SRE1 - expected value of $\hat{\pi}^*$

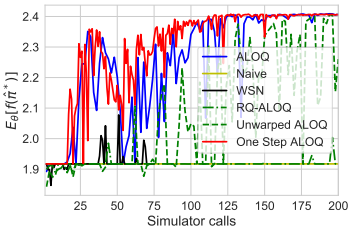


(c) F-SRE1

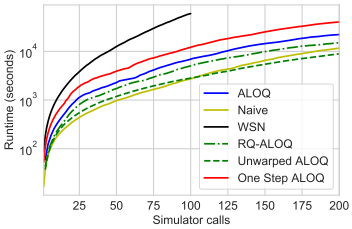
Figure 5: Contour plot of F-SRE1 (values in SRE region have been reduced by a factor of 10), and comparison of performance (higher is better) and runtimes of all methods on F-SRE1.



(a) F-SRE2



(b) F-SRE2 - expected value of $\hat{\pi}^*$



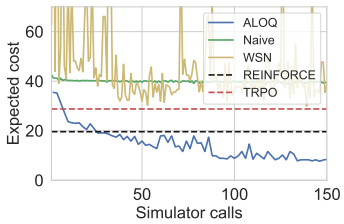
(c) F-SRE2

Figure 6: Contour plot of F-SRE1 (values in SRE region have been reduced by a factor of 10), and comparison of performance (higher is better) and runtimes of all methods on F-SRE2.

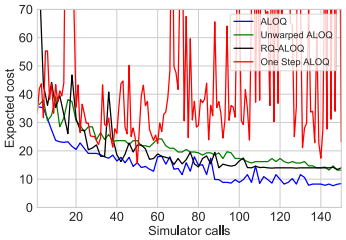
Figures 5a and 6a shows the contour plots of these two functions. Both functions have a narrow band of θ that corresponds to the SRE regions, i.e., the scale of the rewards is much larger in these regions. In F-SRE1 this is $-1 < \theta < 0$ while in F-SRE2 this is $-0.2 < \theta < 0.2$. We downscaled the region corresponding to the SRE by a factor of 10 to make the plots more readable. The final learned policy, i.e., $\hat{\pi}^*$, of ALOQ and the Naïve approach is shown as a vertical line, along with π^* (the true maximum). These lines illustrate that not accounting for SREs properly can lead to learning significantly suboptimal policies.

Figures 5b and 6b, which plot the performance of all methods for the two functions, show that ALOQ substantially outperforms all the other algorithms except for one-step ALOQ (note that both WSN and the naïve approach fail completely in these settings). As expected, intensification does not yield any additional benefit in this low dimensional problem. However, our experiments on the robotics tasks presented next show that intensification is crucial for success in higher dimensional problems.

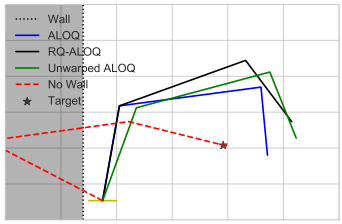
The total runtime is presented in Figures 5c and 6c (note the log scale). Again WSN is significantly slower than all other methods. In fact, it was not computationally feasible to run WSN beyond 100 data points for F-SRE2.



(a) Expected costs of different $\hat{\pi}^*$ - Baselines

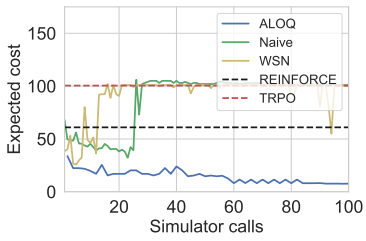


(b) Expected costs of different $\hat{\pi}^*$ - Ablations

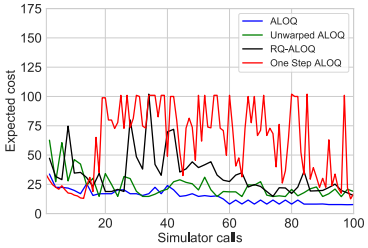


(c) Learned arm configurations

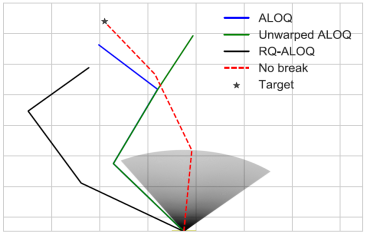
Figure 7: Performance and learned configurations on the robotic arm collision avoidance task. In (a) the performance of REINFORCE and TRPO is at convergence.



(a) Expected costs of different $\hat{\pi}^*$ - Baselines



(b) Expected costs of different $\hat{\pi}^*$ - Ablations



(c) Learned arm configurations

Figure 8: Performance and learned configurations on the robotic arm joint breakage task. In (a) the performance of REINFORCE and TRPO is at convergence.

8.2.2. Robotic Arm Simulator

In the next experiment, we evaluate ALOQ's performance on a robot control problem implemented in a kinematic simulator. The goal is to configure each of the three controllable joints of a robot arm such that the tip of the arm gets as close as possible to a predefined target point.

Collision Avoidance: In the first setting, we assume that the robotic arm is part of a mobile robot that has localised itself near the target. However, due to localisation errors, there is a small possibility that it is near a wall and some joint angles may lead to the arm colliding with the wall and incurring a large cost. Minimising cost entails getting as close to the target as possible while avoiding the region where the wall may be present.

Figures 7a and 7b show the expected cost (lower is better) of the arm configurations after each timestep for each method. ALOQ, unwarped ALOQ, and RQ-ALOQ greatly outperform the other baselines. REINFORCE and TRPO are not sample efficient, exhibiting a slow rate of improvement, while WSN fails to converge at all.

Figure 7c shows the learned arm configurations, as well as the policy that would be learned by ALOQ if there was no wall (No Wall). The shaded region represents the possible locations of the wall. This plot illustrates that ALOQ learns a policy that gets closest to the target. Furthermore, while all the BO based algorithms learn to avoid the wall, active selection of θ allows ALOQ to do so more quickly: smart quadrature allows it to more

efficiently observe rare events and accurately estimate their boundary. For readability we have only presented the arm configurations for algorithms with performance comparable to ALOQ. Note that due to constraints on the joint angles, it is not possible to reach the target while avoiding the region where the wall may be present.

Joint Breakage: To see how ALOQ performs in settings with continuous environment variables, we consider a variation in which instead of uncertainty introduced by localisation, some settings of the first joint carry a 5% probability of it breaking, which consequently incurs a large cost. Minimising cost thus entails getting as close to the target as possible, while minimising the probability of the joint breaking.

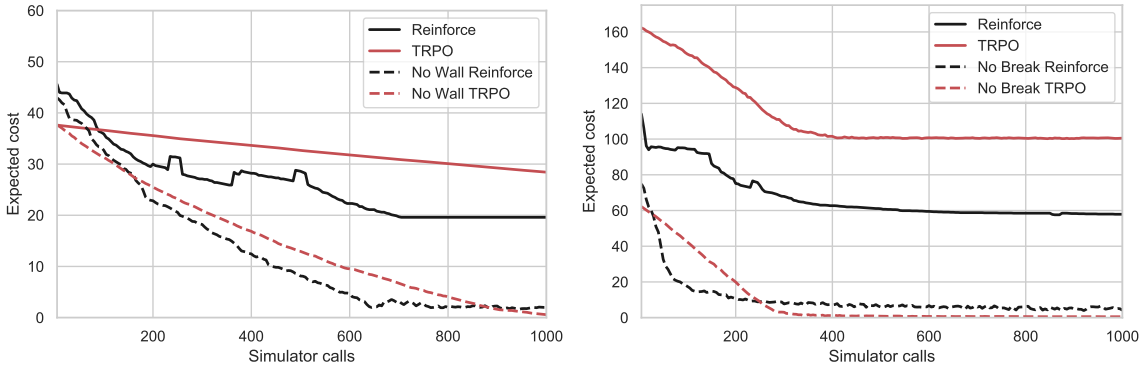
Figures 8a and 8b shows the expected cost (lower is better) of the arm configurations after each timestep for each method. Since θ is continuous in this setting, and WSN requires discrete θ, it was run on a slightly different version with θ discretised by 100 equidistant points. The results are similar to the previous experiment, except that the baselines perform worse. In particular, the Naïve baseline, WSN, and REINFORCE seem to converge to a suboptimal policy since they have not witnessed any SREs.

Figure 8c shows the learned arm configurations together with the policy that would be learned if there were no SREs (No break). The shaded region represents the joint angles that can lead to failure. This figure illustrates that ALOQ learns a qualitatively different policy than the other algorithms, one that avoids the joint angles that might lead to a breakage while still getting close to the target faster than the other methods. Again, for readability we only present the arm configurations for the most competitive algorithms.

Performance of REINFORCE and TRPO: Both REINFORCE and TRPO are relatively sample inefficient. However, one question that arises is whether these methods eventually find the optimal policy. To check this, we ran them for a total of 10000 simulator calls each. We repeated this for both the Collision Avoidance and Joint Breakage settings. To further check if the size of the neural net policy makes a difference, we repeated the TRPO experiment with neural nets with two hidden layers with (5,5), (16,16), and (32,32) units each. Of these three policies, the policy with (32,32) units performed the best in the Collision Avoidance experiment, while the one with (5,5) was the best in the Joint Breakage experiment. The learning curves for these are presented in Figure 9 (we only present the results up to 1000 simulator calls for readability; there is no improvement beyond what can be seen in the plot). Both baselines can solve the tasks in settings without SREs, i.e., where there is no possibility of a collision or a breakage (No Wall and No Break in the figures). However, in settings with SREs, they converge rapidly to a suboptimal policy from which they are unable recover even if run for much longer, since they do not experience the SREs often enough. This phenomenon has been explored further in Paul et al. (2019).

Setting with approximate $p(\theta)$: Now we consider the setting where the exact distribution $p(\theta)$ is not known a priori, but only an approximation is available. In this setting, instead of directly setting the robot arm's joint angles, we set the torque applied to each joint (π). The final joint angles are determined by the torque and the unknown friction between the joints (θ). Setting the torque too high can lead to the joint being damaged or broken, which incurs a large cost.

We use the simulator as a proxy for both real trials as well as the simulated trials. In the first case, we simply sample θ from a uniform prior, run a baseline policy, and use the observed returns to compute an approximate posterior over θ. We then use ALOQ to



(a) Collision avoidance task

(b) Arm breakage task

Figure 9: Performance of REINFORCE and TRPO on the robotic arm simulator.

compute the optimal policy over this posterior ('ALOQ policy'). For comparison, we also compute the *maximum a posteriori* (MAP) estimate of θ and the corresponding optimal policy (MAP policy). To show that active selection of θ is advantageous, we also compare against the policy learned by RQ-ALOQ.

Since we are approximating the unknown $p(\theta)$ with a set of samples, it makes sense to keep the sample size relatively low for computational efficiency when finding the ALOQ policy (50 samples in this instance). However, to show that ALOQ is robust to this approximation, when comparing the performance of the ALOQ and MAP policies, we used a much larger sample size of 400 for the posterior distribution.

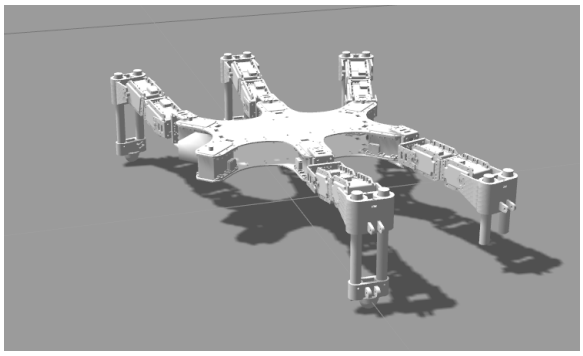
For evaluation, we drew 1000 samples of θ from the more granular posterior distribution and measured the returns of the three policies for each of the samples. The average cost incurred by the ALOQ policy (presented in Table 1) was 31% lower than that incurred by the MAP policy and 23.6% lower than the RQ-ALOQ policy. This is because ALOQ finds a policy that slightly underperforms the MAP policy in some cases but avoids over 95% of the SREs (cost $\geq$70 in Table 1) experienced by the MAP and RQ-ALOQ policies.

	Average Cost	% Episodes in Cost Range 0-20	20-70	$\geq$70
ALOQ Policy	19.82	61.3%	38.5%	0.2%
MAP Policy	28.76	67.1%	28.7%	4.2%
RQ-ALOQ	25.95	-	94.5%	5.5%

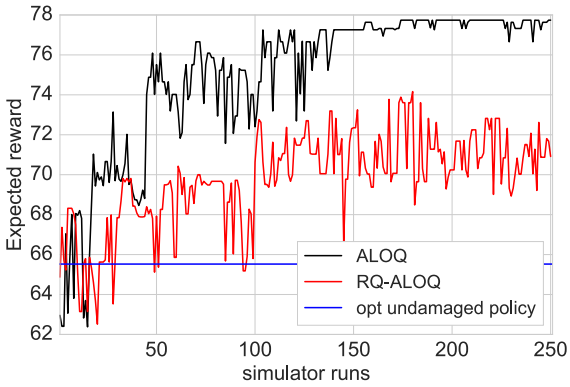
Table 1: Comparison of the performance of ALOQ, MAP, and RQ-ALOQ policies when $p(\theta)$ must be estimated.

8.2.3. Hexapod Locomotion Task

As robots move from fully controlled environments to more complex ones, they have to face the inevitable risk of getting damaged. However, it may be expensive or even impossible



(a) Hexapod with a shortened and a missing leg.



(b) Expected value of $\hat{\pi}^*$

Figure 10: Hexapod locomotion problem.

to decommission a robot whenever any damage condition prevents it from completing its task. Hence, it is desirable to develop methods that enable robots to recover from failure.

Intelligent trial and error (IT&E) (Cully et al., 2015) has been shown to recover from various damage conditions and thereby prevent catastrophic failure. Before deployment, IT&E uses the simulator to create an archive of diverse and locally high performing policies for the intact robot that are mapped to a lower dimensional *behaviour space*. If the robot becomes damaged after deployment, it uses BO to quickly find the policy in the archive that has the highest performance on the damaged robot. However, it can only respond after damage has occurred. Though it adapts quickly, performance will be poor during the initial trials after damage occurs as the initial deployed policy only optimises performance on the undamaged robot. To mitigate this effect, we propose to use ALOQ to learn in simulation the policy with the highest expected performance across the possible damage conditions. By deploying this policy, instead of the policy that is optimal for the intact robot, we can minimise in expectation the negative effects of damage in the period before IT&E has learned to recover.

We consider a hexapod locomotion task with a setup similar to that of Cully et al. (2015) to demonstrate this experimentally. The objective is to cross a finish line a fixed distance from its starting point. Failure to cross the line leads to a large negative reward, while the reward for completing the task is inversely proportional to the time taken.

It is possible that a subset of the legs may be damaged or broken when deployed in a physical setting. For our experiments, we assume that, based on prior experience, any of the front two or back two legs can be shortened or removed with probability of 10% and 5% respectively, independent of the other legs, leading to 81 possible configurations. We excluded the middle two legs from our experiment as their failure has relatively little impact on the hexapod's movement. The configuration of the six legs acts as our environment variable. Figure 10a shows one such setting. The environment variable here has a discrete support with 81 states, one for each setting of the four legs with the corresponding probability for that setting. For example, the probability of the setting {no damage, removed, shortened, no damage} is $0.85 \times 0.05 \times 0.1 \times 0.85$.

We applied ALOQ to learn the optimal policy given these damage probabilities, but restricted the search to the policies in the archive created by Cully et al. (2015).[2] Figure 10b shows that ALOQ finds a policy with much higher expected reward than RQ-ALOQ. It also shows the policy that generates the maximum reward when none of the legs are damaged or broken ('opt undamaged policy' - this is the optimal policy that IT&E would initially deploy).

As mentioned earlier, while IT&E can be used to search for a new policy once some damage occurs, the returns during the initial phase of the search depends heavily on the deployed policy. In this experiment, conditioned on a damage occurring, the ALOQ policy gets an expected return of 77.7 while the optimal policy learned by MAP-Elites gets an expected return of 65.5.

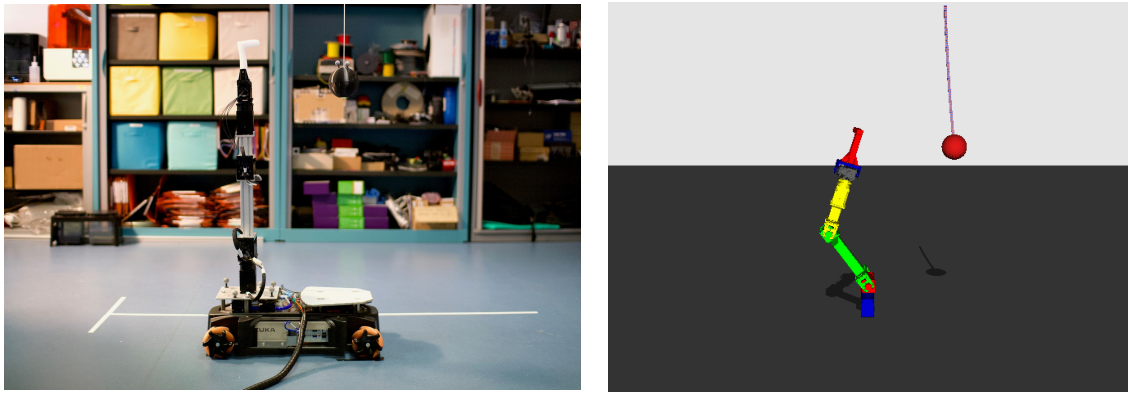
To check if a policy learnt by ALOQ in simulation transfers successfully, we ran an experiment where we used ALOQ to learn a policy entirely in simulation and then deployed the learnt policy on a real hexapod. In order to limit the number of physical trials required to evaluate the ALOQ policy, we limited the possibility of damage to the rear two legs. The learnt policy performed better than the opt undamaged policy (which was also learnt only in simulation) on the physical robot because it optimised performance on the rare configurations that matter most for expected return (e.g., either leg shortened). However, the performance of both policies were worse than in simulation due to the reality gap. For example, in simulation the undamaged hexapod traveled more than $1m$ with both the ALOQ policy and the opt undamaged policy, while in reality it traveled no more than $0.75m$. These results underscore the need for an algorithm like TALOQ to bridge the reality gap. In the next section we present an experiment using TALOQ with a robotic arm without using MAP-Elites.

8.3. TALOQ Experiments

To see how well TALOQ learns transferable policies compared to ALOQ and other baselines, we applied it to a robotic arm. The arm has actuators across 4 joints that control the position of its end effector in 3D space. The objective is to strike a ball hanging from a rope with its end effector and achieve a target velocity. The reward function is the sum of two components: a squared exponential function of the velocity of the ball with a sharp peak at $0.75m/s$, and a cost that increases linearly with the minimum distance of the end effector from the centre of the ball (observed throughout the whole trajectory). The policy space is 5-D consisting of 4 joint angles and a frequency parameter for the arm to oscillate between the initial configuration and the specified joint angles. As an SRE, we assume that the arm is damaged with probability 5%, making the actuator for the third joint unresponsive and stuck in the initial configuration. This can cause policies that perform well on the fully functional arm to perform poorly on the damaged arm.

Since physical trials are too expensive to compare TALOQ against all the other baselines on the physical robot, we first do an extensive evaluation of TALOQ in simulation, and then compare TALOQ to ALOQ in an experiment on the physical robot.

2. These 12514 policies were generated by MAP-Elites (Mouret and Clune, 2015) using a model of the intact robot in simulation using DART (Lee et al., 2018).



(a) Physical robot

(b) Simulated robot

Figure 11: Experimental setup for TALOQ: The objective is to learn a policy for striking the ball and achieving a particular velocity. (a) The physical setup, (b) the setup in simulation.

8.3.1. Simulated Experiments

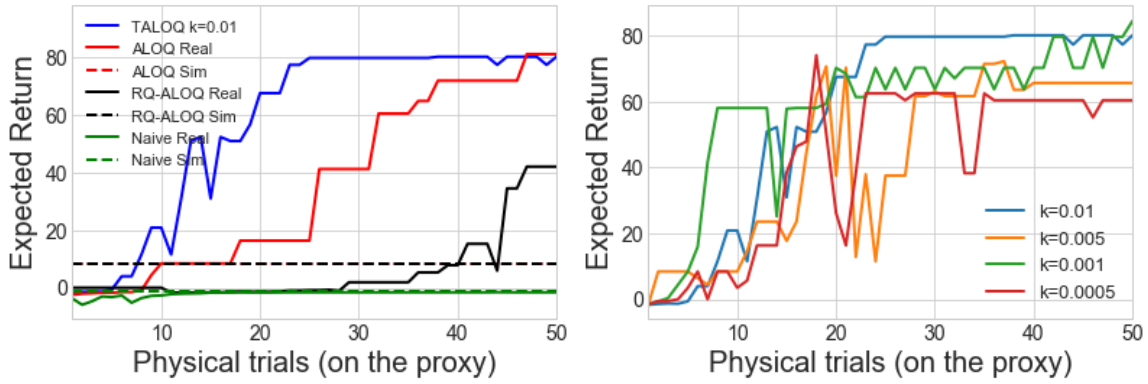
To enable extensive evaluation of TALOQ, we devised a simulated version of the experiment. We used DART (Lee et al., 2018) to design a simulator for our experimental setup.[3] Figures 11a and 11b show the physical robot and the simulated model.

We treated the simulator for the robotic arm as a proxy for reality, and developed another version which we treated as the simulator. To create a reality gap between the two, instead of the ball hanging from the rope we modelled it as a pendulum. We also set the mass of the ball to $1kg$ compared to $60g$ in the proxy. Since simulated trials are relatively cheap, we ran 20 random replicates and after each iteration evaluated the expected return of the policy TALOQ specified as optimal.

In Figure 12a we compare the performance of TALOQ against three baselines: ALOQ, RQ-ALOQ, and the naïve approach. We learnt two versions of each of the baseline, one solely in simulation to check if the learnt policies transfer well, and the other solely on the proxy for reality to check that TALOQ indeed makes use of the simulated trials to improve sample efficiency on the proxy. Note that the x-axis is the number of physical trials. The policy learnt only in simulation using ALOQ (ALOQ Sim)/RQ-ALOQ (RQ-ALOQ Sim)/Naïve (Naïve Sim) approach does not transfer at all to the proxy for reality, since the reality gap is quite large. Learning a policy with ALOQ exclusively in this proxy (ALOQ Real) performs better since there is no reality gap. However it is less sample efficient than TALOQ since it does not leverage the information provided by the simulator. This shows that TALOQ can effectively leverage simulated trials to improve sample efficiency on the physical system. On the other hand, the naïve approach (Naïve Real) does not learn at all due the presence of the SRE.

To see how the performance of TALOQ varies based on the choice of k in (16), we ran also the experiment with different values of k. The results presented in Figure 12b shows that performance is stable across a wide range of k, though higher values can lead to slightly faster learning.

3. We used the robot_dart wrapper: `https://github.com/resibots/robot_dart`.



(a) Comparison of performance

(b) Effect of k on TALOQ's performance

Figure 12: Performance of TALOQ on the simulated experiments.

Physical trials	Replicate # 1	2	3
1	-2.19	-6.07	-4.98
≤ 10	7.99	-6.07	35.85
≤ 20	28.18	-6.07	35.85
≤ 35	28.18	89.18	56.40
≤ 50	28.18	89.18	93.60

Table 2: Evolution of the expected reward on the physical robot using TALOQ

8.3.2. Physical Experiments

Next we used TALOQ to learn a policy that is transferable to the real physical robot using simulated and physical trials. We compare its performance to that of the policy learnt by ALOQ only in simulation. Policies learnt by ALOQ only in simulation with a budget of 200 trials achieved a median return of 4.23 on the robot across 20 random replicates, which shows that it transfers poorly. This is unsurprising since the reality gap can be significant due to the modelling errors and the differences in the controller. Compared to this, across 3 random replicates with a total budget of 200 trials, TALOQ needed 13, 34, and 48 physical trials to find policies with expected returns of 28.18, 89.18, and 93.60 (see Table 2). This demonstrates that combining physical and simulated trials during the learning process using TALOQ learns a much better policy.[4]

9. Conclusions

In this article, we presented ALOQ, a novel approach using BO and BQ to perform sample-efficient RL in a way that is robust to the presence of significant rare events. We also presented TALOQ, an extension to ALOQ that addresses the problem of the reality gap by actively selecting when to evaluate on the physical system instead of the imperfect simulator.

4. A video of one learnt policy is available at `https://youtu.be/R8Ss-dhDCmo`.

We empirically evaluated ALOQ on different simulated tasks involving a robotic arm simulator, and a hexapod locomotion task and showed how it can be also be applied to settings where the distribution of the environment variable is unknown a priori. Our results demonstrated that ALOQ outperforms multiple baselines, including related methods proposed in the literature. Further, ALOQ is computationally efficient and does not require any restrictive assumptions to be made about the environment variables. We also showed that TALOQ can be used to successfully learn a policy that is robust to the SREs while addressing the challenge posed by the reality gap.

Acknowledgments

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreements #637713 and #637972).

Appendix A. Beta Warping

As mentioned earlier, warping the inputs with a Beta CDF can be used to transform non-stationary function into a stationary one. An example of such a transformation is provided in Figure 13.

Appendix B. General Experimental Details

We provide further details of our experiments in this section.

Covariance function: Across all experiments we use a squared exponential covariance function given in (3).

Treatment of hyperparameters: Algorithms 1 and 2 requires all the hyperparameters to be updated after every iteration. While this update can be performed using MLE or MAP, we follow a full Bayesian approach and compute the marginalised posterior distribution $p(f \mid \mathcal{D})$ by first placing a hyperprior distribution on ζ, the set of all hyperparameters (including beta warping parameters and lengthscale δ for TALOQ), and then marginalising it out from $p(f \mid \mathcal{D}, \zeta)$. In practice, an analytical solution for this is unlikely to exist so we approximate it using random samples from the posterior $p(\zeta \mid \mathcal{D})$ drawn using slice sampling (Neal, 2000).

Choice of hyperpriors: We assume a log-normal hyperprior distribution for all the above hyperparameters. For the variance we use $(\mu = 0, \sigma = 1)$, while for the lengthscales we use $(\mu = 0, \sigma = 0.75)$ across all experiments. For the beta warping parameters we used $(\mu = 2, \sigma = 0.5)$ for all artificial test functions, and $(\mu = 0, \sigma = 0.5)$ for the robotic simulator tasks.

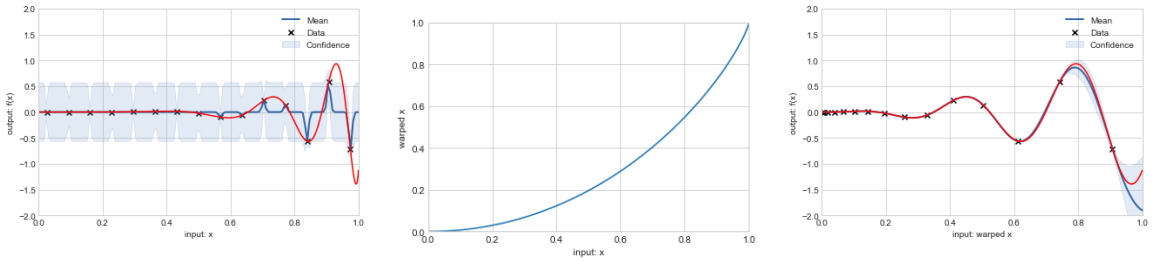
Optimising the BO/BQ acquisition functions: We used DIRECT (Jones et al., 1993) to maximise the BO acquisition function α_{ALOQ}. To minimise the BQ acquisition function, we exhaustively computed $\mathbb{V}(\bar{f}(\pi_{t+1})|\mathcal{D}_{1:t}, \pi_{t+1}, \theta)$ for each θ since this was computationally very cheap.

Appendix C. Robotic Arm Simulator

The configuration of the robot arm is determined by three joint angles, each of which is normalised to lie in $[0, 1]$. The arm has a reach of $[-0.54, 0.89]$ on the x-axis. We set $\kappa = 1.5$ for all three experiments in this section.

C.1. Collision Avoidance

In this setting the environment variable θ is the location of the wall. It is discrete with 20 support points logarithmically distributed in $[-0.2, 0.14]$. The first 10 locations closest to the arm have a probability of 0.75% each, the next 7 have 1.5% each, and final 3 locations furthest from the arm have a probability of 27.33% each. The cost incurred is 100 times the distance between the target and the final position of the end effector, and collisions with the wall yield a cost of 2500.



(a) Fitting a GP to the unwarped function

(b) Warping function

(c) GP fitted to the warped function

Figure 13: Effect of beta warping: (a) A GP (blue line) is fitted to the observations (marked with x) from the original function (red line). Note how it struggles to model the function because of the non-stationary nature. (b) The warping function $\beta(2, 0.8)$ used to transform the input space. It contracts the space closer 0, while stretching it closer to 1. (c) The GP fitted to the transformed observations is able to model the function much better.

C.2. Joint Breakage

Angles between $[0.3, 0.7]$ for the first joint have an associated 5% probability of breakage which leads to a cost of 2000. Otherwise the agent incurs a cost of 100 times the distance between the target and the final position of the end effector. Thus, in this case $p(\theta) \sim \text{Bin}(1, 0.05)$, while during learning ALOQ can set it to $\text{Bin}(1, q)$ where $q \in [0, 1]$.

C.3. REINFORCE and TRPO

For TRPO we used a neural net with two hidden layers with 5 units each, and had a KL constraint of 0.01. For REINFORCE we used a linear Gaussian policy and set the learning rate to 10^{-4}.

C.4. Setting with approximate $p(\theta)$

As described in the paper, in this setting we assume that $\pi \in [0, 1]^3$ is the torque applied to the joints, and $\theta \in [0.5, 1]$ controls the rigidity of the joints. The final joint angle is determined as π/θ. If the torque applied to any of the joints is greater than the rigidity, (i.e. any of the angles end up > 1), then the joint is damaged, incurring a large cost.

To simulate a set of n physical trials with a baseline policy π_b, we sample θ from $U(0.5, 1)$ and observe the return $f(\pi_b, \theta)$ and add iid Gaussian noise to them. The posterior can be computed as $p(\theta|\mathcal{D}^b_{1:n}, \pi_b) \propto p(\theta)p(\mathcal{D}^b_{1:n}|\pi_b, \theta)$, where $\mathcal{D}^b_{1:n} = \{(\pi_b, f_1), (\pi_b, f_2), ..., (\pi_b, f_n)\}$. We can approximate this using slice sampling since both the prior and the likelihood are analytical.

An alternative formulation would be to corrupt the joint angles with Gaussian noise instead of the observed returns. The posterior can still be computed in this case, but instead of using slice sampling, we would have to make use of *approximate Bayesian computation* Rubin (1984); Tavaré et al. (1997), which would be computationally expensive.

To ensure that only the information gained about θ gets carried over from the physical trials to the final ALOQ/MAP policy being learned, the target for the baseline policy was different to the target for the final policy.

As mentioned in the paper, to find the optimal policy using ALOQ, we approximated the posterior with 50 samples using a slice sampler. However, for evaluation and comparison with the MAP policy, we used a much more granular approximation with 400 samples.

Appendix D. Hexapod Locomotion Task

The robot has six legs with three degrees of freedom each. We built a fairly accurate model of the robot which involved creating a URDF model with dynamic properties of each of the legs and the body, including their weights, and used the DART simulator for the dynamic physics simulation.[5] We also used velocity actuators.

The low-level controller (or *policy*) is the same open-loop controller as in Cully and Mouret (2015) and Cully et al. (2015). The position of the first two joints of each of the six legs is controlled by a periodic function with three parameters: an offset, a phase shift, and an amplitude (we keep the frequency fixed). The position of the third joint of each leg is the opposite of the position of the second one, so that the last segment always stays vertical. This results in 36 parameters.

The archive of policies in the behaviour space was created using the MAP-Elites algorithm Mouret and Clune (2015). MAP-Elites searches for the highest-performing solution for each point in the duty factor space Cully et al. (2015), i.e., the time each tip of the leg spent touching the ground. MAP-Elites also acts as a dimensionality reduction algorithm and maps the high dimensional controller/policy space (in our case 36D) to the lower dimensional behaviour space (in our case 6D). We also used this lower dimensional representation of the policies in the archive as the policy search space (π) for ALOQ.

For our experiment, we set the reward such that failure to cross the finish line within 5 seconds yields zero reward, while crossing the finish line gives a reward of $100 + 50v$ where v is the average velocity in m/s.

Appendix E. TALOQ - Physical Experiment

In our experiments, each episode lasted for 4 seconds and the reward function was defined as follows:

$$r = -25 d_{\min} + 100 \exp(-10(v_x - v_{\text{target}})^2) \tag{20}$$

where $d_{\min}$ is the minimum distance between the end-effector and the ball observed throught the episode, v_x is the velocity of the ball (in the x-direction) at the time of the impact with the robotic arm (if there is no impact, $v_x = 0$) and v_{target} is the target velocity. In essence, we want to hit the ball with the arm and generate a velocity of v_{target} in the x-direction. In the simulated experiments (Sec. 8.3.1) we set $v_{\text{target}} = 1\,m/s$, whereas in the physical experiments (Sec. 8.3.2) we set $v_{\text{target}} = 0.75\,m/s$.

5. `https://dartsim.github.io`

References

Thomas Bartz-Beielstein, Christian W. G. Lasarczyk, and Mike Preuss. Sequential parameter optimization. In *IEEE Congress on Evolutionary Computation*, 2005.

François-Xavier Briol, Chris J. Oates, Mark Girolami, Michael A. Osborne, and Dino Sejdinovic. Probabilistic Integration: A Role for Statisticians in Numerical Analysis? *ArXiv e-prints*, 2015.

Eric Brochu, Vlad M Cora, and Nando de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. eprint arXiv:1012.2599, arXiv.org, December 2010.

Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 2015.

Konstantinos Chatzilygeroudis and Jean-Baptiste Mouret. Using Parameterized Black-Box Priors to Scale Up Model-Based Policy Search for Robotics. In *International Conference on Robotics and Automation (ICRA)*, 2018.

Konstantinos Chatzilygeroudis, Roberto Rama, Rituraj Kaushik, Dorian Goepp, Vassilis Vassiliades, and Jean-Baptiste Mouret. Black-Box Data-efficient Policy Search for Robotics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

Konstantinos Chatzilygeroudis, Vassilis Vassiliades, Freek Stulp, Sylvain Calinon, and Jean-Baptiste Mouret. A survey on policy search algorithms for learning robot controllers in a handful of trials. *IEEE Transactions on Robotics*, 2019.

Yevgen Chebotar, Mrinal Kalakrishnan, Ali Yahya, Adrian Li, Stefan Schaal, and Sergey Levine. Path integral guided policy search. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

Kamil Ciosek and Shimon Whiteson. Offer: Off-environment reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2017.

Dennis D. Cox and Susan John. A statistical method for global optimization. In *IEEE International Conference on Systems, Man and Cybernetics*, 1992.

Dennis D. Cox and Susan John. Sdo: A statistical method for global optimization. In *Multidisciplinary Design Optimization: State-of-the-Art*, 1997.

Antoine Cully and Jean-Baptiste Mouret. Evolving a behavioral repertoire for a walking robot. *Evolutionary Computation*, 2015.

Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 2015.

Marc Peter Deisenroth and Carl Edward Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning (ICML)*, 2011.

Jordan Frank, Shie Mannor, and Doina Precup. Reinforcement learning in the presence of rare events. In *International Conference on Machine Learning (ICML)*, 2008.

Peter W. Glynn. Likelihood ratio gradient estimation for stochastic systems. In *Communications of the ACM*, 1990.

Tom Gunter, Michael A. Osborne, Roman Garnett, Philipp Hennig, and Stephen Roberts. Sampling for inference in probabilistic models with fast bayesian quadrature. In *Neural Information Processing Systems (NIPS)*, 2014.

Philipp Hennig, Michael A. Osborne, and Mark Girolami. Probabilistic numerics and uncertainty in computations. *Proceedings of the Royal Society of London A: Mathematical, Physical and*

Engineering Sciences, 2015.

Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, and Kevin P. Murphy. An experimental investigation of model-based parameter optimisation: Spo and beyond. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, 2009.

Nick Jakobi. Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adaptive Behavior*, 1997.

Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In Federico Morán, Alvaro Moreno, Juan Julián Merelo, and Pablo Chacón, editors, *Advances in Artificial Life*, 1995.

Donald R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Applications*, 1993.

Donald R. Jones, Matthias Schonlau, and WilliamJ. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 1998.

Sanket Kamthe and Marc Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. In *International Conference on Artificial Intelligence and Statistics*, pages 1701–1710, 2018.

Motonobu Kanagawa, Bharath K. Sriperumbudur, and Kenji Fukumizu. Convergence guarantees for kernel-based quadrature rules in misspecified settings. In *Neural Information Processing Systems (NIPS)*, 2016.

Sylvain Koos, Jean-Baptiste Mouret, and Stéphane Doncieux. The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Transactions on Evolutionary Computation*, 2013.

Andreas Krause and Cheng S Ong. Contextual gaussian process bandit optimization. In *Neural Information Processing Systems (NIPS)*, 2011.

Jeongseok Lee et al. DART: Dynamic Animation and Robotics Toolkit. *The Journal of Open Source Software*, 2018.

Sergey Levine and Vladlen Koltun. Guided policy search. In *International Conference on International Conference on Machine Learning (ICML)*, 2013.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research (JMLR)*, 2016.

Hod Lipson and Jordan B. Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 2000.

Daniel J. Lizotte, Tao Wang, Michael Bowling, and Dale Schuurmans. Automatic gait optimization with gaussian process regression. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

Alonso Marco, Felix Berkenkamp, Philipp Hennig, Angela P. Schoellig, Andreas Krause, Stefan Schaal, and Sebastian Trimpe. Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with Bayesian optimization. In *International Conference on Robotics and Automation (ICRA)*, 2017.

Ruben Martinez-Cantin, Nando de Freitas, Arnaud Doucet, and José Castellanos. Active policy learning for robot planning and exploration under uncertainty. In *Robotics: Science and Systems*, 2007.

Ruben Martinez-Cantin, Nando de Freitas, Eric Brochu, José Castellanos, and Arnaud Doucet. A bayesian exploration-exploitation approach for optimal online sensing and planning with a visually

guided mobile robot. *Autonomous Robots*, 2009.

Jonas Močkus. On bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, 1975.

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arxiv:1504.04909*, 2015.

Radford Neal. Slice sampling. *Annals of Statistics*, 2000.

Anthony O'Hagan. Monte carlo is fundamentally unsound. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 1987.

Anthony O'Hagan. Bayes-hermite quadrature. *Journal of Statistical Planning and Inference*, 1991.

Michael Osborne, Roman Garnett, Zoubin Ghahramani, David K Duvenaud, Stephen J Roberts, and Carl E Rasmussen. Active learning of model evidence using bayesian quadrature. In *Neural Information Processing Systems (NIPS)*, 2012.

Supratik Paul, Konstantinos Chatzilygeroudis, Kamil Ciosek, Jean-Baptiste Mouret, Michael Osborne, and Shimon Whiteson. Alternating optimisation and quadrature for robust control. In *AAAI Conference on Artificial Intelligence*, 2018.

Supratik Paul, Michael A. Osborne, and Shimon Whiteson. Fingerprint policy optimisation for robust reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2019.

Rémi Pautrat, Konstantinos Chatzilygeroudis, and Jean-Baptiste Mouret. Bayesian optimization with automatic prior selection for data-efficient direct policy search. In *Proceedings 2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

Jan Peters and Stefan Schaal. Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. *CoRR*, abs/1703.02702, 2017.

Matthias Poloczek, Jialei Wang, and Peter Frazier. Multi-information source optimization. In *Neural Information Processing Systems (NIPS)*. 2017.

Aravind Rajeswaran, Sarvjeet Ghotra, Sergey Levine, and Balaraman Ravindran. Epopt: Learning robust neural network policies using model ensembles. *International Conference on Learning Representations (ICLR)*, 2017.

Carl Edward Rasmussen and Zoubin Ghahramani. Bayesian monte carlo. *Neural Information Processing Systems (NIPS)*, 2003.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

Donald B. Rubin. Bayesianly justifiable and relevant frequency calculations for the applied statistician. *The Annals of Statistics*, 1984.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, 2015.

Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 2010.

Jasper Snoek, Kevin Swersky, Rich Zemel, and Ryan Adams. Input warping for bayesian optimization of non-stationary functions. In *International Conference on International Conference on Machine Learning (ICML)*, 2014.

Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: no regret and experimental design. In *International Conference on Machine Learning (ICML)*, 2010.

Simon Tavaré, David J. Balding, R. C. Griffiths, and Peter Donnelly. Inferring coalescence times from dna sequence data. *Genetics*, 1997.

Saul Toscano-Palmerin and Peter I. Frazier. Bayesian Optimization with Expensive Integrands. *ArXiv e-prints*, March 2018.

Brian J. Williams, Thomas J. Santner, and William I. Notz. Sequential design of computer experiments to minimize integrated response functions. *Statistica Sinica*, 2000.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992.

Ali Yahya, Adrian Li, Mrinal Kalakrishnan, Yevgen Chebotar, and Sergey Levine. Collective robot reinforcement learning with distributed asynchronous guided policy search. In *International Conference on Intelligent Robots and Systems (IROS)*, 2017.